A low cost and highly accurate technique for big data spatial-temporal interpolation

Esmaeilbeigi, M., Chatrabgoun, O., Hosseinian-Far, A., Montasari, R. & Daneshkhah, A.

Original citation & hyperlink:
Esmaeilbeigi, M, Chatrabgoun, O, Hosseinian-Far, A, Montasari, R & Daneshkhah, A 2020, 'A low cost and highly accurate technique for big data spatial-temporal interpolation', Applied Numerical Mathematics, vol. 153, pp. 492-502.
https://dx.doi.org/10.1016/j.apnum.2020.03.009

DOI 10.1016/j.apnum.2020.03.009
ISSN 0168-9274

Publisher: Elsevier

.

A Low Cost and Highly Accurate Technique for Big Data Spatial-Temporal Interpolation

M. Esmaeilbeigi[a], O. Chatrabgoun[b], A. Hosseinian-Far[c], R. Montasari[d], and A. Daneshkhah[e,*]

[a]*Department of Mathematics, Faculty of Mathematical Sciences and Statistics, Malayer University, Iran*
[b]*Department of Statistics, Faculty of Mathematical Sciences and Statistics, Malayer University, Iran*
[c]*Department of Business Systems & Operations, University of Northampton, UK*
[d]*School of Computing and Engineering, University of Huddersfield, Queensgate, Huddersfield, HD1 3DH*
[e]*Faculty of Engineering, Environment and Computing, Coventry University, UK*

Abstract

The high velocity, variety and volume of data generation by today's systems have necessitated Big Data (BD) analytic techniques. This has penetrated a wide range of industries; BD as a notion has various types and characteristics, and therefore a variety of analytic techniques would be required. The traditional analysis methods are typically unable to analyse spatial-temporal BD. Interpolation is required to approximate the values between the already exiting data points, yet since there exist both location and time dimensions, only a multivariate interpolation would be appropriate. Nevertheless, existing software are unable to perform such complex interpolations. To overcome this challenge, this paper presents a layer by layer interpolation approach for spatial-temporal BD. Developing this layered structure provides the opportunity for working with much smaller linear system of equations. Consequently, this structure increases the accuracy and stability of numerical structure of the considered BD interpolation. To construct this layer by layer interpolation, we have used the good properties of Radial Basis Functions (RBFs). The proposed new approach is applied to numerical examples in spatial-temporal big data and the obtained results confirm the high accuracy and low computational cost. Finally, our approach is applied to explore one of the air pollution indices, i.e. daily $PM_{2.5}$ concentration, based on different stations in the contiguous United States, and it is evaluated by leave-one-out cross validation.

Keywords: Spatial-temporal interpolation, Big data, Radial basis functions, Layer by layer interpolation.
2010 MSC: 41A05, 65D05.

*Corresponding author
Email addresses: m.esmaeilbeigi@malayeru.ac.ir (M. Esmaeilbeigi), o.chatrabgoun@malayeru.ac.ir (O. Chatrabgoun), amin.hosseinian-far@northampton.ac.uk (A. Hosseinian-Far), r.montasari@hud.ac.uk (R. Montasari), Ali.Daneshkhah@coventry.ac.uk (and A. Daneshkhah)

1. Introduction

The amount of data in today's systems, businesses and society in general is rapidly increasing, and therefore the ability in analysing large datasets (also known as BD) and being able to make informed decisions have become a key basis for competitions, underpinning new waves of productivity growth, innovation and consumer surplus [1]. BD requires a collection of analysis techniques to turn raw data into relevant information. Many of such are advanced mathematical, statistical and computational methods. BD are generated in a wide range of industries,disciplines and applications including but not limited to business transactions, weather data, sensor signals, search engine queries, multimedia materials and social network activities. Within the analysts' community, it is widely accepted that BD can be conceptualized by the three dimensions [2]: Volume, Velocity and Variety. Volume refers to the vast amounts of data being generated and recorded, while velocity refers to the pace of data streaming, that is, the speed at which data are generated, recorded and communicated. Also, variety refers to the heterogeneity of data sources and formats. Veracity is a fourth V that exists in other sources [3]. Despite the fact that big data and large datasets are different concepts, to most people big data implies an enormous volume of numbers, images, videos or texts. Some recent developments and comprehensive studies in big data analysis can be seen in [4, 5, 6]. Spatial-temporal big data refers to the type of BD that has Spatial (location) and temporal (Time) dimensions; instances include data collected from different meteorology stations in different time zones. Although the spatial-temporal big data is widely studied e.g. see [7, 8, 9, 10], one of the key discussions in big data analytics has always been about spatial-temporal interpolation and its complexity [11]. One solution to overcome the complexity of spatial-temporal interpolation is utilise a routine multivariate interpolation. In order to remove dimentionality, radial basis functions (RBFs) are often used, nevertheless there are still two major challenges: 1) Although there are advantages in RBFs i.e. no need for a mesh or triangulation, simple implementation, dimension independence, and no staircasing or polygonization for boundaries, typical interpolation methods can not be applied on BD. Moreover, such calculations are impossible for spatial-temporal big data in the usual way and common application software generate error messages. 2) Mainly linear system of equations derived by RBF's approximation with a high order of convergence [12, 13], are ill-conditioned and unstable [14], and usually include a full interpolant matrix; in other words, this will make the situation even more complex. Solving the system of equations derived by RBFs will have a very high computational cost. Besides, this can lead to an intense instability within the considered challenge itself, as the condition number of the equations system, which is the ill-conditioning criterion, will be very large.

To overcome these concerns and by creating a layered structure for spatial-temporal big data, the interpolation would become practically possible. Such a structure enables the analysis of a much smaller linear system of equations in the presence of big data. In other words, our approach increases the accuracy and stability of numerical structure of the considered spatial-temporal big data interpolation. Developing a layer

by layer interpolation for big data requires the existing information to be in a layered format, and this is common in spatial-temporal data. Hence, the data of the first layer is fixed for the second layer and the data from the first and second layers are fixed for the third layer and so on. Numerical results demonstrate the capabilities and the improved stability and efficiency of the layer-by-layer interpolation approach for the big dataset. The primary contribution of this paper is to present an efficient and flexible algorithm for solving multilayer BD interpolation using a layer by layer interpolation method, and subsequently for spatial-temporal big data. We apply the useful properties of RBFs, such as high order of convergence and ease of implementation, for the layer by layer big data interpolation. More specifically, the method is implemented using Gaussian RBF. This efficient algorithm results in a much smaller and well-condition interpolation matrix.

The remaining sections of the paper are organized as follows: the notion of big data interpolation of an unknown multivariate function is discussed in Section 2. In Section 3, we demonstrate the use of RBFs for function approximation and we further discuss the circumstances in which they are used for multi-layer interpolation of big data problems. Moreover, the methods of layer-by-layer interpolation approach for big data set using RBFs is discussed in Section 3; this is followed by demonstrating its application for spatial-temporal big data problem. The results of numerical experiments for spatial-temporal BD are presented in Section 4. In Section 5, our approach is applied to explore one of the air pollution indices, i.e. daily $PM_{2.5}$ concentration, based on data derived from different stations in the contiguous United States. Please note that the numerical results are obtained using MATLAB/Python programming.

2. Multilayer Interpolation in the Big Data Problems

In this section, we initially define the multilayer big data interpolation challenge and we then consider interpolation in spatial-temporal big data as a special case. In the classic form of interpolation, assume that we have a big data set, $\{X_i\}_{i=1}^N \subset \mathbb{R}^d$, and dependent data values, $\{f_i\}_{i=1}^N \subset \mathbb{R}$. For the desired basis functions (aka blending function) $\{\phi_i\}_{i=1}^N$, we would like to find an interpolant function of the following form:

$$\tilde{f}(X) = \sum_{i=1}^{N} \lambda_i \phi_i(X),$$

such that $\tilde{f}(X_i) = f_i$ is true, for every point in our data set. However, such an interpolation which is also known as classic interpolation is impossible for big data, and due the complexity involved many commonly used application software generate error messages. In practice, the considered big data set $\{X_i\}_{i=1}^N$ may have a layered structure. For instance, in a two layer structure, we have the following form:

$$\{X_i\}_{i=1}^N = \left\{ \left(X_{i_1}^{(1)}, X_{i_2}^{(2)} \right) \right\} \begin{array}{l} i_1 = 1, 2, \ldots, N_1, \\ i_2 = 1, 2, \ldots, N_2, \end{array}$$

where $N = N_1 \times N_2$. Therefore, we can derive the following information:

$$\left(X_{i_1}^{(1)}, X_{i_2}^{(2)}, f_{i_1 i_2}\right); \quad \begin{matrix} i_1 = 1, 2, \ldots, N_1. \\ i_2 = 1, 2, \ldots, N_2. \end{matrix}$$

The data in the first layer are $\left\{X_{i_1}^{(1)}\right\}_{i_1=1}^{N_1}$, which belong to $\mathbb{R}^{d_1}$ and the second layer information consist of $\left\{X_{i_2}^{(2)}\right\}_{i_2=1}^{N_2}$, belong to $\mathbb{R}^{d_2}$. We would like to find an interpolant function,

$$\tilde{f}(X) = \sum_{i_1=1}^{N_1} \sum_{i_2=1}^{N_2} \lambda_{i_1,i_2} \phi_{i_1,i_2}(X),$$

such that $\tilde{f}(X_i) = \tilde{f}(X_{i_1}^{(1)}, X_{i_2}^{(2)}) = f_{i_1 i_2}$, for $i_1 = 1, 2, \ldots, N_1$ and $i_2 = 1, 2, \ldots, N_2$, and also $i = 1, 2, \ldots, N_1 \times N_2$. In the general form of multilayer big data interpolation in M-layers f is a function of independent variables $X^{(1)}$ to $X^{(M)}$, is in the direction of each independent variable, and the scattered big data is selected with appropriate distribution. In fact, we attempt to model and fit the quantity f as a function of the independent variables $X^{(1)}$ to $X^{(M)}$. Therefore, we will have the following information:

$$\left(X_{i_1}^{(1)}, X_{i_2}^{(2)}, \ldots, X_{i_M}^{(M)}, f_{i_1 i_2 \ldots i_M}\right); \quad \begin{matrix} i_1 = 1, 2, \ldots, N_1 \\ i_2 = 1, 2, \ldots, N_2 \\ \vdots \\ i_M = 1, 2, \ldots, N_M \end{matrix} \tag{1}$$

which $\left\{X_{i_1}^{(1)}\right\}_{i_1=1}^{N_1} \in \mathbb{R}^{d_1}$, $\left\{X_{i_2}^{(2)}\right\}_{i_2=1}^{N_2} \in \mathbb{R}^{d_2}$, $\ldots$, $\left\{X_{i_M}^{(M)}\right\}_{i_M=1}^{N_M} \in \mathbb{R}^{d_M}$. We would like to find the interpolant function

$$\tilde{f}(X) = \sum_{i_1=1}^{N_1} \sum_{i_2=1}^{N_2} \cdots \sum_{i_M=1}^{N_M} \lambda_{i_1,i_2,\ldots,i_M} \phi_{i_1,i_2,\ldots,i_M}(X),$$

such that

$$\tilde{f}(X_i) = \tilde{f}(X_{i_1}^{(1)}, \ldots, X_{i_M}^{(M)}) = f_{i_1 \ldots i_M}, \tag{2}$$

for $i_1 = 1, 2, \ldots, N_1$, $i_2 = 1, 2, \ldots, N_2$, by continuing this process until $i_M = 1, 2, \ldots, N_M$, and also $i = 1, 2, \ldots, N_1 \times N_2 \times \ldots \times N_M$. The big data on the first layer have been fixed with respect to the data on the second layer, which means that for any element in $X_{i_2}^{(2)}$, such as $X_K^{(2)}$ where $K \in \{1, 2, \ldots, N_2\}$ and for $i_1 = 1, 2, \ldots, N_1$, we have the following information:

$$\left(X_{i_1}^{(1)}, X_K^{(2)}, f_{i_1 K}\right), \ i_1 = 1, 2, \ldots, N_1.$$

In other layers, the same approach could be repeated. For instance, the big data on the first and the second layers have been fixed with respect to the third layer. Therefore, for any element in $X_{i_3}^{(3)}$, such as $X_S^{(3)}$ where $S \in \{1, 2, \ldots, N_3\}$ and for each $X_{i_1}^{(1)}$, $i_1 = 1, 2, \ldots, N_1$, and for each $X_{i_2}^{(2)}$, $i_2 = 1, 2, \ldots, N_2$, we have

the following information:

$$\left(X_{i_1}^{(1)}, X_{i_2}^{(2)}, X_S^{(3)}, f_{i_1 i_2 S}\right), \;\; i_1 = 1, 2, \ldots, N_1, \; i_2 = 1, 2, \ldots, N_2.$$

As discussed, one of the aims in this paper is to construct layer by layer interpolation for big data based on RBFs, and to apply the useful properties of the RBFs. Further information on RBFs are provided in the Appendix A.

In the following paragraphs, we have tried to apply the useful properties introduced in the RBFs, especially Gaussian RBF, for multilayer interpolation in the presence of big data. For this purpose, assume that f is a function of independent variables $X^{(1)}$ to $X^{(M)}$, is in the direction of each independent variable, and scattered data are selected with appropriate distribution. Sampling of f in these situations has been recorded, and we have tried to model and fit the quantity f as a function of the independent variables $X^{(1)}$ to $X^{(M)}$ as M layers. In other words, we have big data like the one in Equation (1), and we intend to construct the interpolant function f based on the given big data using RBFs. Considering what was discussed earlier, interpolation based on RBFs for given data in (1) can be constructed as follow:

$$\tilde{f}(X^{(1)}, \ldots, X^{(M)}) = \sum_{i_1=1}^{N_1} \cdots \sum_{i_M=1}^{N_M} \lambda_{i_1,\ldots,i_M} \phi_{i_1,\ldots,i_M}(X^{(1)}, \ldots, X^{(M)}), \tag{3}$$

which $\phi_{i_1,i_2,\ldots,i_M}(X^{(1)}, X^{(2)}, \ldots, X^{(M)})$ is the interested RBF with center $({X_{i_1}}^{(1)}, {X_{i_2}}^{(2)} \ldots, {X_{i_M}}^{(M)})$, i.e.

$$\phi_{i_1,i_2,\ldots,i_M}(X^{(1)}, \ldots, X^{(M)}) = \varphi(||(X^{(1)}, \ldots, X^{(M)}) - ({X_{i_1}}^{(1)}, \ldots, {X_{i_M}}^{(M)})||_2).$$

The interpolant (3) in accordance with the interpolation conditions (2) leads to a linear system of equations by $N_1 \times N_2 \times \ldots \times N_M$ equations and $N_1 \times N_2 \times \ldots \times N_M$ unknown values. For the reasons discussed in previous sections, solving this linear system of equations which is derived from RBFs is impossible for big data. The reason is that the linear system of equations is ill-conditioned and unstable, and includes a full and complex interpolation matrix. Furthermore, to solve such system of equations derived by RBFs for big data, we will face a very high computational cost and consequently application software are not capable of computing it without issuing any error messages. Moreover, this can lead to an intense instability within the big data problem under examination. In other words, the condition number of the system of equations, which is an indicator for ill-condition measurement, will be very large. For instance, if we solve the system of equations (2) by the lower-upper (LU) decomposition, the computational cost is proportional to $N_1^3 \times N_2^3 \times \ldots \times N_M^3$, i.e. $O(N_1^3 \times N_2^3 \times \ldots \times N_M^3)$. Similarly, in case of N_1, N_2, $\ldots N_M$, we will encounter a very large amount of computational resource requirements, and the condition number of the system of equations, which is a benchmark for ill-conditioned situation, will be very large. Computing such an operation is practically impossible for big data.

Layer by layer interpolation structure provides the ability for handling a much smaller system of linear equations in big data. It also increases the accuracy and stability of the numerical structure.

As an application of the multilayer interpolation in the spatial-temporal big data, daily $PM_{2.5}$ concentration can be referred to as a function of location and time variables. As a case study, this air pollution index has been a very controversial issue e.g. see [15] and [16]. In fact, information from meteorology stations which have been collected at different times can be considered as a big dataset for multilayer spatial-temporal interpolation. The information can be presented as $(X_{i_1}, t_{i_2}, PM_{i_1 i_2})$ for $i_1 = 1, 2, \ldots, N_1$, and $i_2 = 1, 2, \ldots, N_2$ where $PM_{i_1 i_2}$ is the daily $PM_{2.5}$ concentration in location X_{i_1}, and time t_{i_2}, as the first and the second layer, respectively. A distribution of meteorological stations $\{X_i\}_{i=1}^{N}$, is the first layer, and contains information on $\mathbb{R}^3(/\mathbb{R}^2)$. Besides, it includes distribution of different times $\{t_{i_2}\}_{i_2=1}^{N_2}$ in $\mathbb{R}$ as the second layer, i.e. $(t_{i_2} \in \mathbb{R}, X_{i_1} \in \mathbb{R}^3)$. We can now model the daily $PM_{2.5}$ concentration quantity as a function of location and time. For this purpose, we consider the interpolant function as:

$$\tilde{f}(X,t) = \sum_{i_1=1}^{N_1} \sum_{i_2=1}^{N_2} \lambda_{i_1 i_2} \phi_{i_1,i_2}(X,t),$$

where $\phi_{i_1 i_2}(X,t)$ is the interested RBF with center (X_{i_1}, t_{i_2}), and is:

$$\phi_{i_1,i_2}(X,t) = \varphi(||(X,t) - (X_{i_1}, t_{i_2})||_2).$$

Also note that

$$||(X,t) - (X_{i_1}, t_{i_2})||_2 = \sqrt{(x - x_{i_1})^2 + (y - y_{i_1})^2 + (z - z_{i_1})^2 + (t - t_{i_2})^2},$$

where $X = (x, y, z)$, and $X_i = (x_i, y_i, z_i)$ for $i_1 = 1, 2, \ldots, N_1$. Based on the interpolation conditions, we have

$$\tilde{f}(X_{i_1}, t_{i_2}) = PM_{i_1 i_2} \quad \begin{matrix} i_1=1,2,\ldots,N_1. \\ i_2=1,2,\ldots,N_2. \end{matrix}$$

This leads to the linear system of equations by $N_1 \times N_2$ equations and $N_1 \times N_2$ unknown values. Using LU decomposition to solve such system of equations, the computational cost is proportional to $N_1^3 \times N_2^3$, i.e. $O(N_1^3 \times N_2^3)$. For instance, if we have information of 1000 of times, in the second layer, and 1000 stations, in the first layer, we would be required to solve a system of 1000000 equations, and the required computational cost is $O((10^6)^3)$; this equals to $O(10^{18})$. Due to such resource intensiveness, it would be computationally infeasible, as discussed earlier.

In the next section, we provide a layer by layer structure and an efficient algorithm for big data interpolation which utilizes the available suitable capabilities of RBFs. In fact, with a significant decrease in the size of the system of linear equations for big data, we are able to obtain results with high accuracy and consistent stability using less computational cost.

3. Layer by layer interpolation approach in the Big data problems

As discussed in the previous section, if the dependent quantity in the interpolation problem of big data is a function of independent variables and in the direction of each of them, we have a distribution of scattered

data, and therefore we can approach it as a multilayer interpolation problem in a big dataset. Although, the RBFs are the best tool for interpolation of high-dimensional scattered data, interpolation by RBFs requires solving a very large system of linear equations with full interpolant matrix with a large condition number. Computing such calculations for big data is practically impossible. To tackle these challenges, we present an efficient and flexible algorithm to solve multilayer interpolation problem in big data using a layer by layer structure. We also apply the useful properties of RBFs in the layer by layer approach. The method is fitted to scattered multilayer big data by Gaussian RBF considering its high order of convergence. This efficient algorithm results in a much smaller and well-conditioned interpolation matrix in the big dataset, which despite the high order of convergence obtained by RBFs, performing calculations would be practically viable. The general structure of this approach is explained in the following paragraphs.

Suppose that we have a big dataset of interpolation in M layers, such as those presented in (1). From ${X_1}^{(M)}$ to ${X_{N_M}}^{(M)}$, we consider N_M alternative interpolation subproblems for big dataset as

$$\begin{aligned}
&({X_{i_1}}^{(1)}, {X_{i_2}}^{(2)} \ldots, {X_{i_{M-1}}}^{(M-1)}, X_1^{(M)}, f_{i_1,i_2,\ldots,i_{M-1},1}),\\
&({X_{i_1}}^{(1)}, {X_{i_2}}^{(2)} \ldots, {X_{i_{M-1}}}^{(M-1)}, X_2^{(M)}, f_{i_1,i_2,\ldots,i_{M-1},2}),\\
&\qquad\qquad\vdots\\
&({X_{i_1}}^{(1)}, {X_{i_2}}^{(2)} \ldots, {X_{i_{M-1}}}^{(M-1)}, X_{N_M}^{(M)}, f_{i_1,i_2,\ldots,i_{M-1},N_M}),
\end{aligned}$$

for $i_1 = 1, 2, \ldots, N_1$, $i_2 = 1, 2, \ldots, N_2$, and by continuing this process until $i_{M-1} = 1, 2, \ldots, N_{M-1}$. For N_M interpolation subproblems above,For N_M we consider the following interpolants, respectively:

$$\begin{aligned}
\tilde{f}_{X_1^{(M)}}(X^{(1)}, \ldots, X^{(M-1)}) &= \sum_{i_1=1}^{N_1} \cdots \sum_{i_{M-1}=1}^{N_{M-1}} \lambda_{i_1,\ldots,i_{M-1}} \phi_{i_1,\ldots,i_{M-1}}(X^{(1)}, \ldots, X^{(M-1)}, X_1^{(M)}),\\
\tilde{f}_{X_2^{(M)}}(X^{(1)}, \ldots, X^{(M-1)}) &= \sum_{i_1=1}^{N_1} \cdots \sum_{i_{M-1}=1}^{N_{M-1}} \lambda_{i_1,\ldots,i_{M-1}} \phi_{i_1,\ldots,i_{M-1}}(X^{(1)}, \ldots, X^{(M-1)}, X_2^{(M)}),\\
&\vdots\\
\tilde{f}_{X_{N_M}^{(M)}}(X^{(1)}, \ldots, X^{(M-1)}) &= \sum_{i_1=1}^{N_1} \cdots \sum_{i_{M-1}=1}^{N_{M-1}} \lambda_{i_1,\ldots,i_{M-1}} \phi_{i_1,\ldots,i_{M-1}}(X^{(1)}, \ldots, X^{(M-1)}, X_{N_M}^{(M)}).
\end{aligned}$$

In this case, we shall solve N_M alternative interpolation subproblems for our considered big dataset. Each one leads to a linear system of equations by $N_1 \times N_2 \times \ldots \times N_{M-1}$ equations and $N_1 \times N_2 \times \ldots \times N_{M-1}$ unknown values. On the other hand, the computational cost to solve each one of such system of equations by LU decomposition is proportional to $N_1^3 \times N_2^3 \times \ldots \times N_{M-1}^3$, i.e. $O(N_1^3 \times N_2^3 \times \ldots \times N_{M-1}^3)$. This is not practically feasible due to its high computational cost. Within our approach, instead of solving the linear system of equations by $N_1 \times N_2 \times \ldots \times N_M$ equations and $N_1 \times N_2 \times \ldots \times N_M$ unknown values, we solve N_M linear systems of equations by $N_1 \times N_2 \times \ldots \times N_{M-1}$ equations and $N_1 \times N_2 \times \ldots \times N_{M-1}$ unknown values. Ultimately, this alternative structure requires the computational cost to be proportional to $N_M \times O(N_1^3 \times N_2^3 \times \ldots \times N_{M-1}^3)$. Since in all of the N_M interpolation subproblems for the considered big dataset, we deal with the same set of data positions i.e. $({X_{i_1}}^{(1)}, {X_{i_2}}^{(2)} \ldots, {X_{i_{M-1}}}^{(M-1)})$, for $i_1 = 1, 2, \ldots, N_1$,

$\ldots, i_{M-1} = 1, 2, \ldots, N_{M-1}$, and due to the radial property of RBFs (that is the interpolation matrix only related to the distance between the interpolation points), we would have the same interpolation matrix in each of the N_M interpolation subproblems. Therefore, instead of solving the system of equations N_M times with the LU decomposition method, we obtain the LU decomposition of the interpolation matrix only once; and this will be similarly used in all systems. In fact, the computational cost is proportional to $O(N_1^3 \times N_2^3 \times \ldots \times N_{M-1}^3)$, as opposed to being $N_M \times O(N_1^3 \times N_2^3 \times \ldots \times N_{M-1}^3)$; This will practically enable the computation. As already discussed, instead of solving a very large linear system of equations which is not viable for big data, we will deal with a number of smaller systems. Moreover, based on the useful properties of RBFs, we apply the LU decomposition for one small interpolation matrix. Nevertheless, the resulting interpolants are not global, and we have access to a corresponding interpolant to each data item in the last layer. In order to express the obtained interpolants in global form (similar to what exists in the classic interpolation by RBFs), solving the following interpolation problem at the end of the process is required:

$$({X_i}^{(M)}, \tilde{f}_{{X_i}^{(M)}}(X^{(1)}, X^{(2)}, \ldots, X^{(M-1)})), \quad i = 1, 2, \ldots, N_M,$$

in which corresponding interpolant is constructed as:

$$\tilde{f}(X^{(1)}, X^{(2)}, \ldots, X^{(M)}) = \sum_{j=1}^{N_M} \lambda_j \phi_j(X^{(M)}),$$

and $\phi_j(X^{(M)})$ is defined as:

$$\phi_j(X^{(M)}) = \varphi(||X^{(M)} - {X_j}^{(M)})||_2).$$

These lead to the solution of the following linear system of equations:

$$\sum_{j=1}^{N_M} \lambda_j \phi_j({X_i}^{(M)}) = \tilde{f}_{{X_i}^{(M)}}(X^{(1)}, X^{(2)}, \ldots, X^{(M-1)}), \quad i = 1, \ldots, N_M.$$

Solving it by LU decomposition results in a computational cost proportional to $O(N_M^3)$. Hence, the final computational cost equals to $O(N_1^3 \times N_2^3 \times \ldots \times N_{M-1}^3) + O(N_M^3)$. Finally, due to dominance of the first term that is obvious for big data, the computational cost equals to $O(N_1^3 \times N_2^3 \times \ldots \times N_{M-1}^3)$, while the computational cost of the resulted classic interpolation by RBFs is proportional to $O(N_1^3 \times N_2^3 \times \ldots \times N_M^3)$, and therefore computation will be practically impossible in the big data context.

Continuity of the reduction process is also possible in other layers, i.e. if we require the computational cost to be proportional to $O(N_1^3 \times N_2^3 \times \ldots \times N_{M-2}^3)$, it is sufficient to use the above-mentioned process in the last layer and the layer prior to it. In the following, we will discuss this in more details.

At this stage, assume that we have the M layers of big data as defined in (1), for (X_1^{M-1}, X_1^M) to

$(X^{M-1}_{N_{M-1}}, X^{M}_{N_{M-1}})$; moreover, we consider $N_M \times N_{M-1}$ alternative interpolation subproblems as:

$$\begin{aligned}
&(X_{i_1}{}^{(1)}, X_{i_2}{}^{(2)} \ldots, X_{i_{M-2}}{}^{(M-1)}, X_1^{(M-1)}, X_1^{(M)}, f_{i_1,i_2,\ldots,i_{M-2},1,1}) \\
&\qquad\qquad \vdots \\
&(X_{i_1}{}^{(1)}, X_{i_2}{}^{(2)} \ldots, X_{i_{M-2}}{}^{(M-1)}, X_{N_{M-1}}^{(M-1)}, X_{N_M}^{(M)}, f_{i_1,i_2,\ldots,i_{M-2},N_{M-1},N_M}),
\end{aligned} \tag{4}$$

for $i_1 = 1, 2, \ldots, N_1, \ldots, i_{M-2} = 1, 2, \ldots, N_{M-2}$. We also consider the $N_M \times N_{M-1}$ interpolation subproblems in (4), and the following interpolants, respectively:

$$\begin{aligned}
\tilde{f}_{X_1^{(M-1)},X_1^{(M)}}(X^{(1)}, \ldots, X^{(M-2)}) &= \sum_{i_1=1}^{N_1} \cdots \sum_{i_{M-2}=1}^{N_{M-2}} \lambda_{i_1,\ldots,i_{M-2}} \phi_{i_1,\ldots,i_{M-2}}(X^{(1)}, \ldots, X^{(M-2)}, X_1^{(M-1)}, X_1^{(M)}), \\
&\vdots \\
\tilde{f}_{X_{N_{M-1}}^{(M-1)},X_{N_M}^{(M)}}(X^{(1)}, \ldots, X^{(M-2)}) &= \sum_{i_1=1}^{N_1} \cdots \sum_{i_{M-2}=1}^{N_{M-2}} \lambda_{i_1,\ldots,i_{M-2}} \phi_{i_1,\ldots,i_{M-2}}(X^{(1)}, \ldots, X^{(M-2)}, X_{N_{M-1}}^{(M-1)}, X_{N_M}^{(M)}).
\end{aligned}$$

We are then required to solve $N_{M-1} \times N_M$ interpolation subproblems for our big dataset. Each one leads to a linear system of equations by $N_1 \times N_2 \times \ldots \times N_{M-2}$ equations and $N_1 \times N_2 \times \ldots \times N_{M-2}$ unknown values. Using the LU decomposition to solve such system of equations, the computational cost is proportional to $N_1^3 \times N_2^3 \times \ldots \times N_{M-2}^3$, i.e. $O(N_1^3 \times N_2^3 \times \ldots \times N_{M-2}^3)$. In fact, instead of solving the linear system of equations consist of $N_1 \times N_2 \times \ldots \times N_M$ equations and $N_1 \times N_2 \times \ldots \times N_M$ unknown values, we would be required to solve $N_{M-1} \times N_M$ linear systems of equations system of $N_1 \times N_2 \times \ldots \times N_{M-2}$ equations and $N_1 \times N_2 \times \ldots \times N_{M-2}$ unknown values. This is also required for the computational costs to be proportional to $N_{M-1} \times N_M \times O(N_1 \times N_2 \times \ldots \times N_{M-2})$. Since in all $N_{M-1} \times N_M$ interpolation problems in (4), we deal with the same set of data positions i.e. $(X_{i_1}{}^{(1)}, X_{i_2}{}^{(2)} \ldots, X_{i_{M-2}}{}^{(M-2)})$, for $i_1 = 1, 2, \ldots, N_1, \ldots, i_{M-2} = 1, 2, \ldots, N_{M-2}$, and due to the radial property of RBFs (that the distance between the interpolation points affects constructing the interpolation matrix), we will only deal with just one analogue matrix in each of the $N_{M-1} \times N_M$ interpolation problems. Therefore, instead of solving the equations $N_{M-1} \times N_M$ times with the LU decomposition method, we obtain the LU decomposition of interpolation matrix only once and use it in all systems. The computational cost instead of $N_{M-1} \times N_M \times O(N_1^3 \times N_2^3 \times \ldots \times N_{M-1}^3)$ is proportional to $O(N_1^3 \times N_2^3 \times \ldots \times N_{M-2}^3)$. This results in significant reductions in the computational cost which is very high for the big dataset. In general, instead of solving a very large linear system of equations, we deal with a number of smaller linear system of equations. Furthermore, based on the above-mentioned properties of RBFs and the fixed positions of the interpolation points, we require the LU decomposition of only one small interpolation matrix rather than a very large original interpolation matrix. The resulting interpolants are in the layer form, and for each of the data items from $(X_1^{(M-1)}, X_1^{(M)})$ to $(X_{N_{M-1}}^{(M-1)}, X_{N_M}^{(M)})$, we access the relevant interpolation. If we wish to express the obtained interpolation seamlessly, it would be sufficient to solve the following interpolation problem:

$$(X_i^{(M-1)}, X_j^{(M)}, \tilde{f}_{X_i^{(M-1)},X_j^{(M)}}(X^{(1)}, X^{(2)}, \ldots, X^{(M-2)}),$$

for $i = 1, 2, \ldots, N_{M-1}$ and $j = 1, 2, \ldots, N_M$.

Moreover, the corresponded interpolation is constructed as:

$$\tilde{f}(X^{(1)}, X^{(2)}, \ldots, X^{(M)}) = \sum_{i=1}^{N_{M-1}} \sum_{j=1}^{N_M} c_{ij} \phi_{ij}(X^{(M-1)}, X^{(M)}),$$

where $\phi_{ij}(X^{(M-1)}, X^{(M)})$ is defined as:

$$\phi_{ij}(X^{(M-1)}, X^{(M)}) = \varphi(||(X^{(M-1)} - X^{(M)}) - ({X_i}^{(M-1)} - {X_j}^{(M)})||_2),$$

and the interpolation problem assists with solving the following linear equations:

$$\sum_{s=1}^{N_{M-1}} \sum_{k=1}^{N_M} c_{sk} \phi_{sk}({X_i}^{(M-1)}, {X_j}^{(M)}) = \tilde{f}_{{X_i}^{(M-1)}, {X_j}^{(M)}}(X^{(1)}, X^{(2)}, \ldots, X^{(M-2)}),$$

for $i = 1, 2, \ldots, N_{M-1}$ and $j = 1, 2, \ldots, N_M$. Furthermore the computational cost is proportional to $O(N_{M-1}^3 N_M^3)$. That is, the final computational cost equals to

$$O(N_1^3 \times N_2^3 \times \ldots \times N_{M-2}^3) + O(N_{M-1}^3 N_M^3). \tag{5}$$

Finally, with regards to the dominance, the computational cost equals to $O(N_1^3 \times N_2^3 \times \ldots \times N_{M-2}^3)$, while the computational cost of the resulted classic method is proportional to $O(N_1^3 \times N_2^3 \times \ldots \times N_M^3)$. The above structure can be continued and can be extended to further one or two layers. In the presence of big data, this structure will increase the efficiency of the method, as the first sentence in (5) is dominant to the second sentence in (5). For a better understanding of this topic, we have provided the following example.

Suppose that the number of layers in the interpolation problem is 4 ($M = 4$), and the number of data items in each layer is 100000 ($N_1 = \ldots = N_4 = 100000$). Then, the computational cost of the above structure (by considering the last two layers) is proportional to $O((10^5)^3 \times (10^5)^3)$, i.e. $O(10^{30})$. While in the classical structure of the RBF interpolation the computational cost is proportional to $O((10^5)^3 \times (10^5)^3 \times (10^5)^3 \times (10^5)^3)$, i.e. $O(10^{60})$. This represents a significant advantage of the proposed algorithm compared to classical ones.

In the next section, using numerical examples, we demonstrate that the algorithm presented in this article can be used for big data, especially spatial-temporal BD.

4. Numerical results in spatial-temporal big data

The results presented in the tables have been obtained in MATLAB/Python on a laptop with a 1.6 GHz Intel Core i7 processor. In order to elaborate on the advantages of the layer by layer interpolation method, numerical results are presented for a big data interpolation problem in two layers which fits well to the spatial-temporal big dataset. Assume that our considered dataset is:

$$(X_i, t_j, f_{ij}) \quad \begin{matrix} i=1,2,\ldots,\mathcal{N}, \\ j=1,2,\ldots,\mathcal{M}, \end{matrix} \tag{6}$$

such that $X_i \in \mathbb{R}^2$ or $X_i \in \mathbb{R}^3$ i.e. the dataset $\{X_i\}_{i=1}^{\mathcal{N}}$ is reported in two or three dimensions, respectively. In fact, we intend to implement the layer by layer interpolation method under the influence of important advantages of Gaussian RBFs. Given the presence of the shape parameter in the Gaussian RBF that is used in the presented numerical results, its calculation should be taken into account, since the accuracy and stability of the numerical results is strongly influenced by choosing a different shape parameter. The optimal shape parameter obtained using the trial and error method can be the most optimal choice [17]. Reporting small CPU time can well present the high capabilities of the new method in terms of computational cost for our considered spatial-temporal big dataset. Moreover, the calculated condition number (Cond) indicates that the new method is very effective in reducing the system of equations that is ill-conditioned. The target of our numerical analysis is thus a two-fold: on one hand, analyzing the efficiency expressed in terms of CPU times; and on the other hand, verifying its accuracy and stability in terms of root mean square (RMS) errors and condition numbers. To illustrate the claims presented above, consider the following numerical examples.

Example 1. We consider layer by layer big data interpolation in two layers, spatial-temporal big data, as defined in (6) such that $X_i \in \mathbb{R}^2$. Layer by layer interpolation with different number of data points in the second layer, $\mathcal{M} = 1000, 10000, 100000$, for the following test function:

$$f(x, y, t) = t^2x^2 + y^2 - 2xyt, \quad 0 \leq x, y, t \leq 1,$$

such that the generated available data using uniform points distribution in each layer are selected. Numerical results for $\mathcal{N} = 64, 81, 100$ points which is the number of points in the first layer are implemented. For instance, for $\mathcal{M} = 100000$ and $\mathcal{N} = 100$, we have 10^7 data points which in fact demonstrates a big data set. Our results, in terms of root mean square (RMS) error, computational time in minutes (Time) and condition number (Cond) are presented in Table 1. These results demonstrate the high capabilities of the layer by layer approach while the classic method is no longer able to perform such calculations by which the system generates the "Out of memory " message when applied in MATLAB/Python programs. This demonstrates the inability and insufficiency of classic methods for calculating spatial-temporal big data sets.

			Layer by Layer		
$\mathcal{M}$	$\mathcal{N}$	c	RMS	Time	Cond
1000	64	0.1	3.08×10^{-3}	0.16	2.28×10^{7}
	81	0.2	3.11×10^{-5}	0.27	2.61×10^{15}
	100	0.5	1.70×10^{-6}	0.39	1.24×10^{19}
10000	64	0.1	2.16×10^{-1}	0.31	4×10^{4}
	81	0.01	2.37×10^{-4}	0.57	2.3×10^{11}
	100	0.01	7.80×10^{-5}	0.66	2.79×10^{17}
100000	64	0.1	2.01×10^{-1}	0.71	4×10^{2}
	81	0.01	2.89×10^{-4}	0.93	2.30×10^{11}
	100	0.1	5.6×10^{-6}	1.24	4.10×10^{17}

Table 1: Layer by layer interpolation for spatial-temporal big data, for different $\mathcal{M}$ and uniform points distribution ($\mathcal{N}$) based on the root mean square error (RMSE), CPU time (in minutes) and condition number (Cond) when $X_i \in \mathbb{R}^2$.

Example 2. In this example, we similarly consider the layer by layer big data interpolation in two layers, spatial-temporal big dat (when $X_i \in \mathbb{R}^3$. Layer by layer interpolation for different number of points in the first) and second layer for spatial-temporal big data is followed. Only in this example, the test function has been changed to the Sinc function. This function is defined as:

$$f(x, y, z, t) = sinc(x)sinc(y)sinc(z)sinc(t), \quad 0 \leq x, y, z \leq 1, \quad 0 \leq t \leq 1,$$

such that

$$sinc(x) = \begin{cases} 1 & x = 0, \\ \frac{sin(\pi x)}{\pi x} & x \neq 0. \end{cases}$$

Note that the generated available data using uniform points distribution in each layer are selected for $\mathcal{N} = 125, 216, 343, 512$ points in the first layer and $\mathcal{M} =$ "1000", "10000", "100000" in the second layer as presented in Table 2. As presented in Table 2, the output demonstrate similar results as previously stated in Example 1. Moreover, we can also consider another different points distribution such as Chebyshev and Halton points. As these distributions will not have much effects on the numerical results, the layer by layer big data interpolation will also work well with these functions. In fact, the same results are taken into account with another points distribution.

			Layer by Layer		
$\mathcal{M}$	$\mathcal{N}$	c	RMS	Time	Cond
1000	125	2	3.05×10^{-4}	0.18	6.2×10^{6}
	216	0.5	1.63×10^{-5}	0.33	3.01×10^{18}
	343	0.6	4.6×10^{-7}	0.56	6.3×10^{19}
	512	0.85	1.1×10^{-8}	0.91	9.98×10^{19}
10000	125	1.5	2.3×10^{-4}	0.37	7.7×10^{7}
	216	0.45	1.2×10^{-5}	0.58	3.4×10^{17}
	343	0.5	1.6×10^{-6}	0.96	8.4×10^{18}
	512	0.65	2.5×10^{-7}	1.22	9.1×10^{19}
100000	125	1.5	3.01×10^{-3}	0.57	1.08×10^{4}
	216	1.5	1.72×10^{-4}	1.38	7.79×10^{7}
	343	0.5	1.21×10^{-5}	1.78	3.01×10^{18}
	512	0.65	1.12×10^{-7}	2.11	1.12×10^{19}

Table 2: Layer by layer interpolation, for different $\mathcal{M}$, Sinc test function and uniform points distribution ($\mathcal{N}$) based on root mean square error (RMSE), CPU time (in minutes) and condition number (Cond) when $X_i \in \mathbb{R}^3$.

5. Application

In this section, the proposed layer by layer interpolation method is applied to determine the daily $PM_{2.5}$ concentrations as a spatial-temporal big data based on different stations. The data are obtained from U.S. Environmental Protection Agency (EPA). Particulate Matter (PM) is the term used to describe condensed phase (solid or liquid) particles suspended in the atmosphere. A growing body of research have focused towards the smaller particles, in particular PM less than 2.5 μm in diameter, as this has been a metric more closely associated with adverse health effects [11]. $PM_{2.5}$ is the mass concentration of airborne particles with an aerodynamic diameter of less than 2.5 μm, expressed in $\mu g/m^3$, where the volume of air is its volume at ambient conditions. The size of 2.5 μm was chosen because of its significance for the penetration of human lungs. This paper focuses on layer by layer interpolation approach for daily $PM_{2.5}$ concentrations as a spatial-temporal big data.

To assess the amount of air pollution over the course of a year in the contiguous United States, daily $PM_{2.5}$ concentration data is used as the experiment of this paper. Our dataset stores daily $PM_{2.5}$ measurements that were obtained from U.S. Environmental Protection Agency (EPA). It contains 146,125 $PM_{2.5}$ concentration measurements collected at 955 monitoring sites on 365 (i.e., $\mathcal{M}$=365) days in 2012. This dataset has the following attributes: day, x, y, and $PM_{2.5}$ concentration measurement, where x and y ($X_i \in \mathbb{R}^2 = (x_i, y_i)$) are the longitude and latitude coordinates of the monitoring sites. The locations of the monitoring sites are illustrated as red dots in Figure 1.

It is noteworthy that we have computed a real world problem and unlike the examples given in the previous section, we have no test function. Therefore, we use leave-one-out cross validation [11] to evaluate the layer by layer interpolation for our spatial-temporal big data. In fact, by performing layer by layer interpolation method on this big data set and putting some points as the evaluation points, we calculated RMS error for the layer by layer method as 1.82×10^{-2}, while the traditional methods failed computationally. The computational time required for the layer by layer interpolation method was 2.17 minutes. Furthermore, the shape parameter c was selected here 0.1.

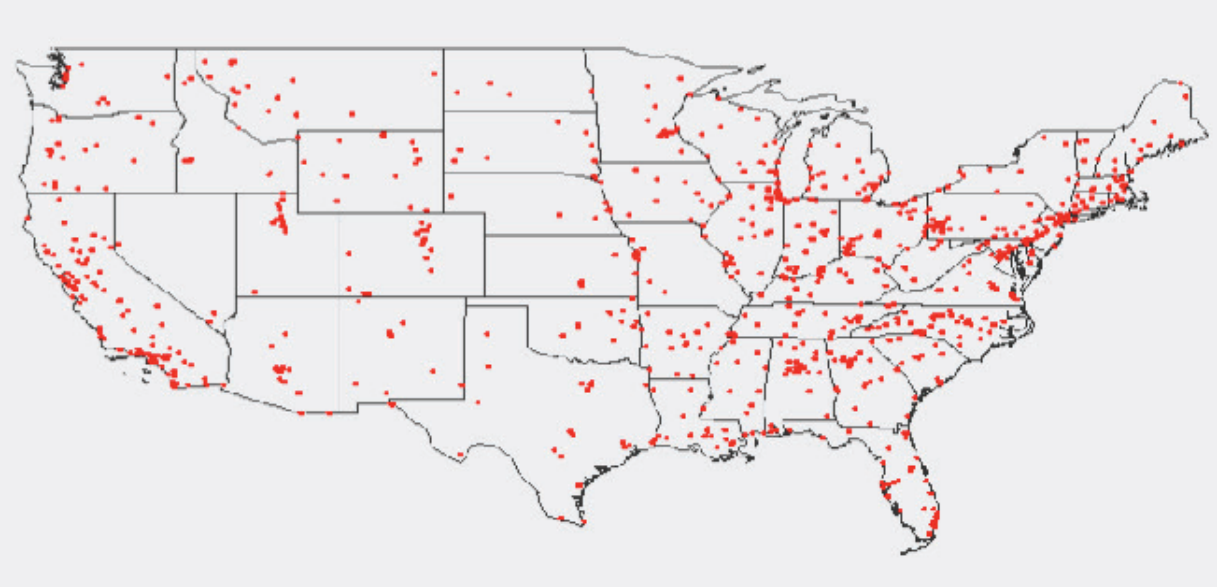

Figure 1: 955 monitoring sites with $PM_{2.5}$ concentration measurements [11].

6. Conclusions

In this paper, we addressed the computational burden of the multivariate interpolation problem using the standard RBFs for the spatial-temporal big data. However, the standard RBFs is very demanding for the multivariate interpolation of the usual datasets and can be implemented in the common software, but the computational cost of using this model for the big data is considerably high, and one would receive a 'low memory' message whilst attempting to implement it using the common software. In this paper, we proposed a layer-by-layer interpolation approach for spatial-temporal big data which is computationally more efficient and the data is effectively handled in a multi-layered process. As discussed above, by creating a layer structure, the computational cost was reduced, and the condition number was decreased. In other words, if the number of layers required for the interpolation problem is M, and the number of data points in the i^{th} layer is N_i, the computational cost of the interpolation problem using the proposed method (i.e., layer-by-layer structure) is proportional to $O(N_{M-1}^3 \times N_M^3)$, while the computational cost of addressing this problem using the classical structure of RBF is proportional to $O(N_1^3 \times N_2^3 \times \ldots \times N_{M-1}^3 \times N_M^3)$. This represents a significant advantage of the proposed algorithm compared to classical one.

The proposed structure would also result in computing a much smaller linear system of equations which would make processing the big spatial-temporal data more tractable. Using the layer by layer RBF approach, based on the last layer and the layer before it, we only need to solve $N_{M-1} \times N_M$ interpolation problems for the big dataset where each one leads to a linear system consists of $N_1 \times N_2 \times \ldots \times N_{M-2}$ equations with $N_1 \times N_2 \times \ldots \times N_{M-2}$ unknown values. As a result, the computational cost of solving the linear system of

equations of size $N_1 \times N_2 \times \ldots \times N_M$ with $N_1 \times N_2 \times \ldots \times N_M$ unknown values will be reduced to solving the linear system of $N_1 \times N_2 \times \ldots \times N_{M-2}$ equations with $N_1 \times N_2 \times \ldots \times N_{M-2}$ unknown values. It will thus increase the accuracy and stability of the numerical computation of the interpolation problem of the big data. Indeed, the proposed method ensures the existence and uniqueness of the solution for the multilayer interpolation of the big data problem.

Acknowledgments

This research was partially supported by funding from the UK Engineering & Physical Sciences Research Council (Strategic Package: Centre for Predictive Modelling in Science and Engineering - Grant No. EP/L027682/1) is acknowledged

References

[1] J. Manyika, M. Chui, B. Brown, J. Bughin, R. Dobbs, C. Roxburgh, A. H. Byers, *Big data: The next frontier for innovation, competition, and productivity*, US: McKinsey Global Institute, (2011).

[2] D. Laney, *3D data management: Controlling data volume, velocity, and variety.*, US: META Group, (2001).

[3] A. Hosseinian-Far, M. Ramachandran, C. Slack, *Emerging Trends in Cloud Computing, Big Data, Fog Computing, IoT and Smart Living*, Strategic Engineering for Cloud Computing and Big Data Analytics, (2017), 29-40.

[4] S. Liu, J. Mcgree, Z. Ge, Y. Xie, *Computational and Statistical Methods for Analysing Big Data with Applications.*, Elsevier Ltd., (2016).

[5] A. Hosseinian-Far, M. Ramachandran, D. Sarwar, *Strategic Engineering for Cloud Computing and Big Data Analytics.*, Springer., (2017).

[6] R. Shukla, J. Agrawal, S. Sharma, G. Singh Tomer, *Data, Engineering and Applications* ., Springer., (2019).

[7] C. Winkle, S. Holan, *Recent Advances in Spatio-Temporal Methodology.*,SR Editorial, Journal of Times Series Analysis, 40 (3), Wiley, (2019).

[8] S. Buteaua, M. Hatzopoulou, D. L. Croused, A. Smargiassi, R. T. Burnett, T. Logani, L. D. Cavellin, M. S. Goldberg, *Comparison of spatiotemporal prediction models of daily exposure of individuals to ambient nitrogen dioxide and ozone in Montreal, Canada.*, Environmental Research, 156, (2017), 201-230.

[9] L. Li, P. Revesz, *Interpolation methods for spatio-temporal geographic data.*, Computers, Environment and Urban Systems, 28, (2004), 201–227.

[10] I. Hussain, G. Spöck a, J. Pilz a, H. Yu, *Spatio-temporal interpolation of precipitation during monsoon periods in Pakistan*, Advances in Water Resources, 33, (2010), 880–886.

[11] T. Losser, L. Li, R. Piltner, *A Spatiotemporal Interpolation Method Using Radial Basis Functions for Geospatiotemporal Big Data*, 2014 Fifth International Conference on Computing for Geospatial Research and Application.

[12] A.H.-D. Cheng, M.A. Golberg, E.J. Kansa, G. Zammito, *Exponential convergence and h-c multiquadric collocation method for partial differential equations*, Numer. Methods Partial Differential Eq., 19 (5), (2003), 571-594.

[13] M. Buhmann, N. Dyn, *Spectral convergence of multiquadric interpolation*, Proc. Edinb. Math. Soc. (2), 36 (2), (1993), 319-333.

[14] G. E. Fasshauer, *Hermite interpolation with radial basis functions on spheres*, Adv. in Comp. Math., 10, (1999), 81-96.

[15] F. Liang, M. Gao, Q. Xiao, G. R. Carmichael, X. Pana, Y. Liuc, *Evaluation of a data fusion approach to estimate daily PM2.5 levels in North China*, Environmental Research, 158, (2017), 54-60.

[16] H. Zhang, Z. Wang, W. Zhang, *Exploring spatiotemporal patterns of PM2.5 in China based on groundlevel observations for 190 cities*, Environmental Pollution, 216, (2016), 559-567.

[17] R. Cavoretto, A. De Rossi, *Spherical interpolation using the partition of unity method: an efficient and flexible algorithm*, Applied Mathematics Letters, 25, (2012), 1251-1256.

[18] G. E. Fasshauer, *Meshfree approximation methods with MATLAB*, Interdisciplinary Mathematical Sciences, vol. 6. World Scientific Publishing Company, Singapore, (2007).

[19] H. Wendland, *Scattered data approximation*, Cambridge Monographs on Applied and Computational Mathematics, 17, Cambridge University Press, Cambridge, (2005).

[20] F.M.B. Martìnez, *Meshless methods for elliptic and free-boundary problems*. PhD thesis, (2008).

[21] G.B. Wright, *Rdial Basis Function Interpolation: Numerical and Analytical Developments*. PhD thesis, (2003).

[22] J. Yoon, *Spectral approximation orders of radial basis function interpolation on the Sobolev space* SIAM J. Math. Anal., 23, (2001), 946-958.

[23] R. Schaback, *Error estimates and condition numbers for radial basis function interpolation*, Adv. Comput. Math., 3, (1995), 251-264.

[24] W. R. Madych, S. A. Nelson, *Bounds on multivariate polynomials and exponential error estimates for multiquadric interpolation*, J. Approx. Theory, 70, (1992), 94-114.

[25] B. Fornberg , N. Flyer, *Accuracy of radial basis function interpolation and derivative approximations on 1-D infinite grids*, Adv. Comput. Math. 23, (2005), 5-20.

[26] W. R. Madych, *Miscellaneous error bounds for multiquadric and related interpolants*, Comput. Math. Appl. 24, (1992), 121-138.

[27] Narcowich FJ, Ward JD. Norm estimates for the inverses of a general class of scattered-data radial-function interpolation matrices. Journal of Approximation Theory. 1992 Apr 1;69(1):84-109.

Appendix A: Radial basis Function

In the interpolation of the scattered data using RBFs ([18]) the approximation of a function $u(X)$ at the centers $\chi = \{X_1, \ldots, X_N\}$, may be written as a linear combination of N RBFs; usually it takes the following form:

$$s_{u,\chi}(X) = \sum_{j=1}^{N} \alpha_j \phi(X - X_j) + \sum_{k=1}^{Q} \beta_k p_k(X). \tag{7}$$

Here, Q denotes the dimension of the polynomial space $\pi_{m-1}(\mathbb{R}^d)$, $p_1, \ldots, p_Q$ denote a basis of $\pi_{m-1}(\mathbb{R}^d)$, $X = (x_1, x_2, \ldots, x_d)$, d is the dimension of the problem, α's and β's are coefficients to be determined, ϕ is the RBF. Some well-known RBFs are listed in Table 3. To cope with additional degrees of freedom, the interpolation conditions

$$s_{u,\chi}(X_j) = u(X_j), \qquad 1 \leq j \leq N, \tag{8}$$

are completed by the additional conditions

$$\sum_{j=1}^{N} \alpha_j p_k(X_j) = 0, \qquad 1 \leq k \leq Q. \tag{9}$$

Solvability of the system is therefore equivalent to solvability of the system

$$\begin{pmatrix} A_{\phi,\chi} & P \\ P^T & 0 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} u\,|_\chi \\ 0 \end{pmatrix}, \tag{10}$$

where $A_{\phi,\chi} = (\phi(X_j - X_k)) \in \mathbb{R}^{N\times N}$ and $P = (p_k(X_j)) \in \mathbb{R}^{N\times Q}$. This last system is obviously solvable if the coefficient matrix on the left-hand side is invertible [19]. Equation (7) can be written without the additional polynomial $\sum_{k=1}^{Q} \beta_k p_k(x)$. In that case, ϕ must be unconditionally positive definite to guarantee the solvability of the resulting system (e.g. Gaussian or inverse multiquadrics). However $\sum_{k=1}^{Q} \beta_k p_k(X)$ is required when ϕ is conditionally positive definite, i.e. when ϕ has a polynomial growth towards infinity. For instance, suppose ϕ is thin plate splines. It must be noted that radial functions that are conditionally positive definite of order one (such as the multiquadric) can be used without appending the constant term to solve the scattered data interpolation problem. Moreover, since positive definite or conditionally positive definite functions are usually globally supported, the interpolation matrix is full and may be very ill-conditioned for some RBFs. Although to improve the conditioning of the system of collocation equations compactly supported RBFs (CSRBFs) have been applied, but the CSRBFs are vanish beyond a user defined threshold distance σ. Therefore, only the entries in the collocation matrix corresponding to collocation nodes lying closer than σ to a given CSRBF center are nonzero, leading to a sparse matrix. In fact, the interest in CSRBFs waned as it became evident that, in order to obtain a good accuracy, the overlap distance σ should cover most nodes in the point set, thus resulting in a populated matrix again [20].

Table 3: Some well-known functions that generate RBFs

Name of function	Definition
Multiquadrics (MQ)	$\phi(x) = \sqrt{\|x\|_2^2 + c^2}$
Inverse multiquadrics (IMQ)	$\phi(x) = \left(\sqrt{\|x\|_2^2 + c^2}\right)^{-1}$
Gaussian (GA)	$\phi(x) = exp\left(-c\|x\|_2^2\right)$
Thin plate splines (TPS)	$\phi(x) = (-1)^{k+1}\|x\|_2^{2k}\log\|x\|_2$
Conical splines	$\phi(x) = \|x\|_2^{2k+1}$

In the present paper, we have used the Gaussian RBFs in our method. The reason is that the Gaussian RBF interpolant has been shown to exhibit "super-spectral" convergence. The accuracy and the stability for the infinitely smooth $\phi(x)$ depend on the number of data points and the value of the shape parameter c [21]. For a fixed c, as the number of data points increase, the RBF interpolation converges to the underlying (sufficiently smooth) function being interpolated at a spectral rate, i.e. $O(e^{-\frac{const.}{h}})$ where h is a measure of the "typical" distance between data points [22, 23, 24]. In certain special cases, such as a Gaussian RBF,

the RBF interpolant has been shown to exhibit "super-spectral" convergence, i.e. $O(e^{-\frac{const.}{h^2}})$ [23, 24, 25]. In either case, the value of *const.* in the estimates is effected by the value of c. For a fixed number of data points, Madych [26] has shown that the accuracy of RBF interpolant can often be significantly improved by decreasing the value of c. However, decreasing c or increasing the number of data points has a severe effect on the stability of the linear system (10). For a fixed c, the condition number of the matrix in the linear system grows exponentially as the number of data points is increased. For a fixed number of data points, as the shape parameter becomes small the condition number of the linear system grows [27].